BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Problem Image Mailbox.

371-25 applic't copy Digest of Papers - 1981 International Test Conference, IEEE. pp. 208-216

SELF-TESTING BY POLYNOMIAL DIVISION

Dilip K. Bhavsar
(315) 456-2746

Richard W. Heckelman
(315) 456-3067

General Electric Company
Electronics Laboratory
Syracuse, NY 13221

ABSTRACT

The concept of division by polynomials in feedback shift registers (FSRs) is extended for self-testing of VLSI. Some fundamentals of FSRs relevant to their usage as test pattern generators and test response compressors in digital testing are revieved. Various design factors and design trade-offs affecting the implementation of the technique at different levels of assembly are discussed. Problems associated with fault-coverage evaluation are highlighted.

1. INTRODUCTION

The advent of VLSI technology and increasing complexity of VLSI chips and systems will make mandatory the inclusion of some form of test feature at all levels of assembly. Two distinct forms of the test features known today are the Design for (external) Test (DFT) and Self-test. The DFT approach includes such techniques as LSSD, Scan-path, and Scan/set [1]-[3] which employ chained serial register paths to simplify the task of testing by external test equipment. These techniques have been successful in ameliorating the test problems for the present level of network complexities. As systems are put on chips and these chips become cheaper, the time and cost of testing by DFT techniques and external automatic test equipment (ATE) will become excessive. Self-testing will also become complex as the functions grow in complexity, but its implementation will only impact the hardware costs. Since the hardware costs continue to decline, the self-test approach is expected to keep pace with semiconductor technology.

1.1 Requirements of Self-testing

The self-testing can be either on-line or off-line. The off-line self-testing has three principal requirements:

1. A source of test inputs (data, clocks, and controls).

2. A means of evaluating test responses.

3. Control of the test procedures, and test status reporting.

When the self-testing is at the lowest level of assembly, i.e., self-testing of a single function, the last requirement generally involves receiving of appropriate start and stop self-test commands. But, when self-testing involves an assembly of several self-testing functions, dedicated hardware may be needed to fulfill this requirement.

There are many ways in which the first two requirements can be met. For example, the test vector source can be a built-in ROM, a counter or an algorithmic pattern generator such as feedback shift register (FSR). The response evaluator can employ a built-in ROM to store expected test responses, or could employ any of the recursive data compressors such as sum checks, transition counting, or signature analysis with feedback shift registers. This paper explores in detail the use of feedback shift registers to meet requirements of both the test vector source and the response evaluator.

A feedback shift register, whether used as a test vector source or a test response evaluator, performs the basic operation of division by a polynomial in GF(2) [4]. For this reason, the self-testing schemes employing FSRs are called "POLYDIV self-testing" schemes or simply POLYDIV.

1.2 Elements of Feedback Shift Register

A linear feedback shift register (Figure 1.1) is a finite state machine consisting of interconnections of essentially three elements, namely, memory elements (delay flip-flops), modulo-2 summers (exclusive-or gates), and binary constant multipliers. (Multiply by 1 implies a connection

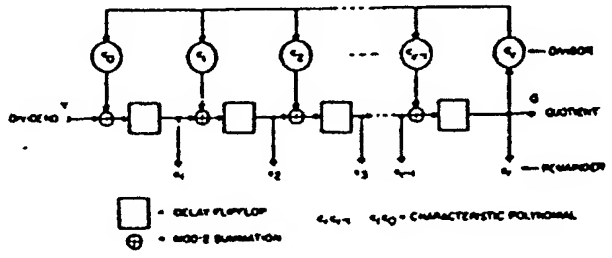

FIGURE 1.1 Feedback Shift Register for Polynomial Division ("PDR")

and by 0 implies absence of a connection). The feedback is applied through the constant multipliers. The constants are actually coefficients of the polynomial $c_r x^r + c_{r-1} x^{r-1} + \cdots + c_1 x^1 + c_0$

(1)

371/22.3
371/22.4

"signature analysis"
dual-mode FSR acts as both test vector gen. and test

where $A = 3(2^m-1)^2/2^r$

r = number of stages in FSR

m = number of FSR bits used in parallel

p = period of PN-sequence

ℓ = lag or delay

In maximal length seqeences, the choice of characteristic polynomial has no effect on the autocorrelation functions which depend only on the number of stages and the number of FSR bits used in parallel. The sequence and the period in non-maximal sequences depend on the starting seed. Their autocorrelation functions, therefore, need analysis of actual sequence.

The PN-sequence generator eliminates the need for deterministic test vector generation and test vector storage. However, it is difficult to predict the effectiveness, i.e., the length of PN-sequence required for achieving desired fault coverage [7]. For a given function under test, this may be affected by a choice of the characteristic polynomial and the seed. The best tool available today for evaluating a PN-sequence is the fault-simulation programs. For the same level of confidence in testing, the random test set is usually longer than the deterministic test set. This increases the cost of fault simulation, but the cost of test generation is not incurred.

2.2 Data Compression in FSRs

The main motivation for using data compression in digital testing is to reduce the amount of test response data to be handled and hence the cost of fault detection. In using FSRs for data compression, the probability of error escape and its effect on fault coverage are the greatest concerns.

Serial Data Compression: A data stream at the serial input of FSRs in Figures 1.1 and 1.2 is recursively compressed into its specific signature. This signature is the last remainder of dividing the input stream by the characteristic polynomial. The error detection by FSRs is based on the fact that if the input stream has one or more bits in error, it is likely to leave a different remainder than the no-error remainder. This error detection is relatively independent of the length of input sequence L and is primarily dependent on the register length r. The error coverage (i.e., the fraction of all the error combinations detected by the compressor) is

$$1 - \frac{2^{L-r} - 1}{2^L - 1}$$

For long input sequences, the error coverage becomes $1 - 2^{-r}$.

Parallel Data Compression: In applications where the channel width is greater than one, the data compression would require a parallel-to-serial converter to use the serial compressor described above. This would be slow and cumbersome. Alternatively, to speed up the compression, several serial compressors (one for each bit in the channel) can be used. However, due to the multitude of FSRs and their signatures, this alternative is not economical.

The parallel compressor shown in Figure 2.1 provides economical and speedy compaction of

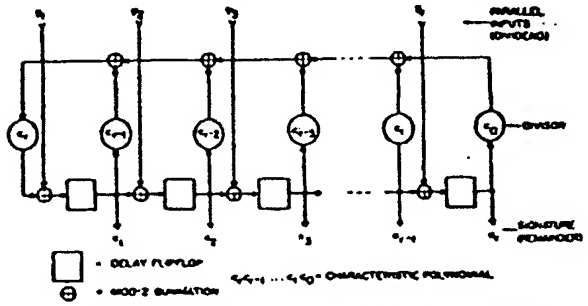

FIGURE 2.1 FSR for Parallel Data Compression

parallel data. Its error coverage is

$$1 - \frac{2^{mL-r} - 1}{2^{mL} - 1}$$

In limiting the case when mL becomes large, the error coverage becomes $1 - 2^{-r}$, which is the same as for the serial compressor.

The process of parallel compression is difficult to visualize readily. One way to describe it is as follows.

The data compression of a sequence of L vectors of r bits by an r-bit parallel compressor is equivalent to superposition of L many time-staggered polynomial division processes, each initialized and triggered by a parallel input to the compressor. The final signature, therefore, is composed of the modulo-2 summation of remainders from all the division processes. This equivalence and the derivation of the error coverage are explained in the Appendix.

Due to the speed and harduare economy, the parallel data compressors are favored in digital testing. The choice of characteristic polynomial has no effect on error coverage, but may affect the fault coverage. Also, the error coverage cannot be taken as a direct measure of effectiveness of the compressor because errors due to faults are not equally likely and not independent. One approach in [8] characterizes classes of dependent errors that are likely to occur and develops error models to determine the likelihood of their detection. But, the method does not give the complete measure as all potential errors are not considered. The best available measure today is the fault coverage as determined by fault simulation. This is discussed in Section 5.

(or simply $c_r c_{r-1} \cdots c_1 c_0$). This polynomial is called the characteristic polynomial of the FSR. The transition from state to state in the FSR is equivalent to the process of division by the characteristic polynomial. The dividend is shifted in via the serial input of the FSR (m.s.b. first), and the quotient appears serially at the output from the right-most stage of FSR. The remainder after each division step is in the content of the FSR.

Figure 1.2 shows an alternative form of FSR which also performs the division by the characteristic polynomial $c_0x^r + c_1x^{r-1} + \cdots + c_{r-1}x + c_r$ ($c_0 c_1 \cdots c_{r-1} c_r$). The quotient obtained is the

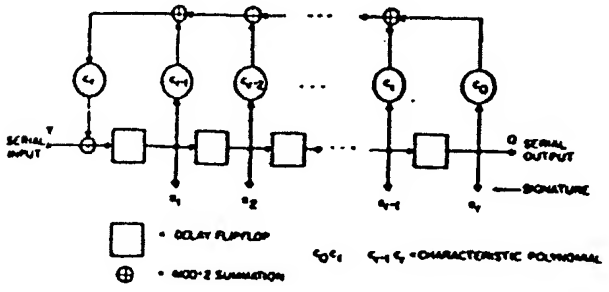

FIGURE 1.2 An Alternative Feedback Shift Register ("FSR")

same as before, but the content of the FSR does not represent the remainder of the division process. Both implementations perform the division by polynomial, but when one needs to be distinguished from the other, the former has been called a Polynomial Divider Register or simply a "PDR."

Both the FSR implementations compress the serial input stream into a word equal in length to the number of stages in the FSR. This is the basis for its use as a data compressor in digital testing. Also, if the FSRs are initialized to a non-zero value and the serial input is tied to a constant (usually a zero), the autonomous behavior of the FSR generates a pseudorandom binary cycle of states which can be used as a test vector set.

In the following sections, the use of FSRs and polynomial division is analyzed in detail. Emphasis is on on-chip application, but the technique is equally applicable for other levels of assembly. Wherever the differences in trade-offs and application techniques arise, they will be mentioned.

2. FSRs IN DIGITAL TESTING

2.1 PN-Sequence Generation in FSRs

In digital testing, two attributes of pseudo-random number sequences (PN-sequence) are important, namely, the sequence period or length, and the statistical properties. They are summarized below.

Sequence Length [4][5]: The length or period of a PN-sequence is determined by the number of stages r in the FSR and its characteristic polynomial. (The degree of characteristic polynomial equals the number of stages in FSR.) When the characteristic polynomial is a primitive, the FSR generates a maximal length sequence. If the characteristic polynomial is irreducible, but non-primitive, then the period of the sequence is a factor of 2^r-1. The period of a non-maximal sequence depends on the starting value called seed. A list of irreducible polynomials of degree 2 through 34 is given in [4].

Statistical Properties†: All maximal length sequences satisfy the following randomness properties [5].

1. In every period, the number of 1's nearly equals the number of -1's.

$$\left| \sum_{k=1}^{p} x_k \right| \le 1$$

2. In every period, half the runs (runs are bursts of 1's or -1's in the PN-sequence) have length one, one-fourth have length two, one-eighth have length three, etc., as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are equally many runs of 1's and -1's.

3. The autocorrelation function $R_x(\ell)$ is two-valued.

$$R_x(\ell) = \frac{1}{p} \sum_{k=1}^{p} x_k \cdot x_{k+\ell}$$

$$= 1 \qquad \ell = 0 \pmod p$$

$$= -\frac{1}{p} \qquad \ell \neq 0 \pmod p$$

4. The above autocorrelation function relates to the serial binary sequence output from any of the FSR stages. When m serial bits of an r stage FSR are used together to form a binary word, the following relationship determines the autocorrelation for the m-bit binary words [6].

$$Ry(\ell) = \frac{2^{2m-\ell} - 2^{\ell} - A}{2^{2m} - 1 - A} \qquad 0 \le \ell \le m-1$$

$$= -\frac{A}{2^{2m} - 1 - A} \qquad m \le \ell < \frac{p}{2}$$

† While doing statistical analysis, it is conventional to assume that binary sequences take on values +1's and -1's instead of 1's and 0's.

3. REQUIREMENTS OF POLYDIV SELF-TESTING

Figure 3.1 shows general features of a POLYDIV self-testing scheme. The function to be tested is

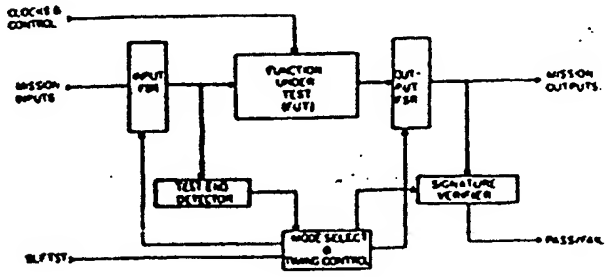

FIGURE 3.1 General POLYDIV Self-Testing Scheme

augmented with linear feedback shift registers on its inputs and outputs. During normal mission operation, these FSRs become transparent, or become intermediate latches, providing registered flow between inputs and outputs. On receiving the self-test command (SLFTST), the Mode Select & Timing Control (MSTC) unit initializes the latches in the two registers to preselected states. The INPUT FSR then provides the test vectors to all the inputs except the clock and control inputs. The OUTPUT FSR recursively compresses the test responses. The TEST-END DETECTOR raises the TSTEND signal to indicate the completion of the test run when a predetermined test-end pattern is generated in the INPUT FSR. At this time the final content (signature) of the OUTPUT FSR is verified by the SIGNATURE VERIFIER by comparing against the expected signature. The expected signature itself is predetermined and hardwired in the SIGNATURE VERIFIER. If an error is detected, an alarm is raised via the PASS/FAIL signal. The function reenters the Mission mode on removal of the SLFTST command.

The FSRs provide a new test vector and compress a new test response in each clock interval. The test rate, therefore, is the system clock rate. Consequently, the clock rate can be increased or decreased to evaluate performance margins. Stored data bursts (test vectors and expected responses) are not required, nor is external automatic test equipment (ATE) required.

Implementation of POLYDIV is usually straightforward. As apparent from Figure 3.1, the POLYDIV does not affect the design of the basic function. But, it does affect the function's interface/interconnection with other functions. This impacts the choices in design of FSRs. Moreover, the success of POLYDIV testing itself is impacted by the basic function type and its design. The level of assembly further affects the design of POLYDIV. Some of these problems and their solutions are discussed in the following section.

4. DESIGN FACTORS IN POLYDIV SELF-TESTING

4.1 FSRs and Interface Dependence

Since the function to be tested is sandwiched between two FSRs, it may be important to make sure that the FSRs do not introduce excessive delays or additional clock intervals in the function's normal operation. There are several alternatives in implementing FSRs. Some of them are discussed below.

Of the two forms of feedback shift registers (Section 1.2), "FSR" is preferred over "PDR" for implementation because the feedback does not introduce exclusive-or gates between the register stages. For the same reason, the register can be easily operated as a part of a chained serial register (CSR) path. A section of FSR consisting of two stages is shown in Figure 4.1. Each stage

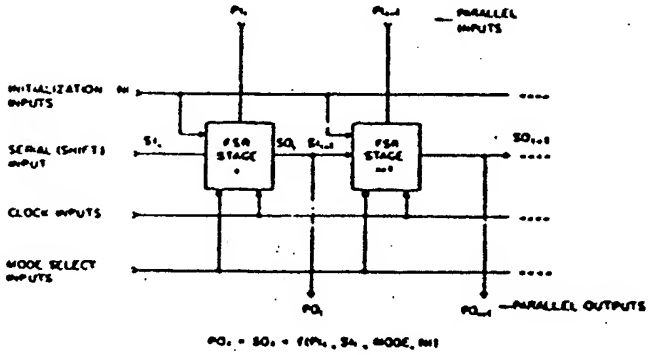

FIGURE 4.1 A Section of FSR Used in POLYDIV

in FSR performs all or some of the following functions.

Initialize Mode: SO = PO = Initial Value

Mission Mode : SO = PO = PI

Generate Mode : SO = PO = SI

Compress Mode : SO = PO = SI ⊕ PI

Each FSR stage (referred to as a cell) consists of a master/slave flip-flop (DFF) and a data select-and-manipulate logic. These two parts of the FSR can be configured in two different ways. Figure 4.2(a) shows an in-stream generator cell. The master/slave DFF in the cell is in the mission data

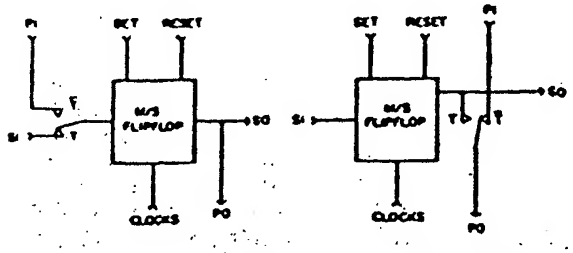

FIGURE 4.2 In-stream and Out-stream Generator Cells

path and is shared by the mission and the test modes. This type of cell is used whenever the function has its own registers or latches at its inputs. The only additional hardware required is in the data select-and-manipulate logic.

Figure 4.2(b) shows an out-stream generator cell. The master/slave DFF in this configuration is completely bypassed when the function is in mission mode. This type of cell is used when the function to be tested is without any registers or latches of its own at its inputs. Use of in-stream cells in this type of function may interfere with the timing and clocking control of the function.

The master/slave DFF itself can be either static or dynamic and can be built in several ways. Implementation of dynamic DFF takes fewer transistors than the static DFF. POLYDIV generally does not require static flip-flops.

The implementation of a COMPRESSOR FSR is similar. Any of the generator cells given in Figure 4.2 can be converted to a compressor cell by feeding an exclusive-or function of PI and SI into the serial input of the generator cell.

Depending on the test and design goals, the FSR may only generate, or compress, or do both. A multi-mode FSR implemented in BILBO [9] operates either as a generator or a compressor, or provides a chained serial register path, or initializes itself. Application of multi-mode FSRs is discussed later in this section.

4.2 Channel Width Dependence

The number of register stages in FSRs is usually kept equal to the number of data bits in the function's input/output channels. But, the function's complexity and its I/O channel widths may force the number of register stages to be different.

Small Input/Output Channel: Maximum period for a PN-sequence is 2^m-1, where m is the number of input bits. For some sequential functions, 2^m-1 test inputs may be too few to achieve the desired fault coverage. A solution here is to extend the test length by feeding m inputs from an extended r-stage FSR ($r > m$).

The data compression at a small output channel faces a similar problem. An FSR with a small number of stages may have unacceptable error escape probability. The solution is similar. The FSR length can be increased to achieve desired error coverage. The unused parallel compressor bits may be tied to logic levels zero or one (Figure 4.3).

Large Input/Output Channel: When the input channel width is large, it may not be necessary to apply all the input combinations as test vectors. For example, a 32-bit FSR may give over four billion patterns, but only a few hundred of the patterns may be sufficient to achieve the desired fault coverage. The solution requires the use of a truncated PN-sequence or a short sequence

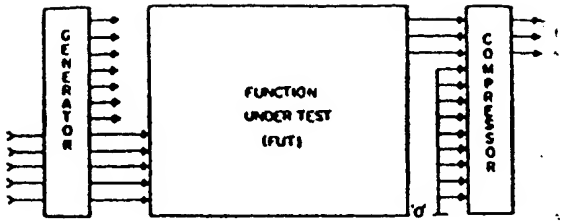

FIGURE 4.3 POLYDIV of Function With Small I/O Channels

resulting from a factorable characteristic polynomial.

The shortening of a PN-sequence affects the randomness properties. However, our greater concern is its impact on the fault coverage. A fault simulation is required to ascertain the adequacy of the fault coverage.

As the error coverage of a compressor increases with its number of stages, larger output channels are not likely to cause concern.

4.3 Indeterminate Signatures

Races and Hazards in Function: Races and hazards are fatal for testing by POLYDIV. An example is shown in Figure 4.4(a). The normal function operation may never get into this situation, but the test operation may due to pseudorandom test inputs. The simulation of POLYDIV in such cases quickly fills up the compressor with don't knows ('x's) reflecting an indeterminate signature. The POLYDIV, therefore, requires that the function be designed to perform correctly regardless of circuit rise time, delay time, or fall time.

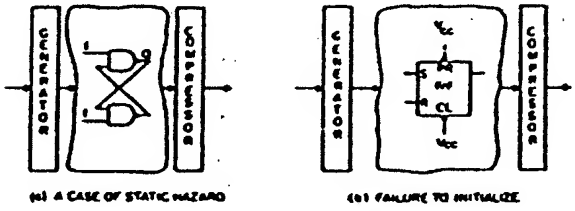

FIGURE 4.4 Indeterminate Signatures

Failure to Initialize: Another case which causes indeterminate signatures is shown in Figure 4.4(b). The basic function is highly sequential and has non-combinational initialization (i.e., the master clear and preset are not available). If the POLYDIV initializes the FSRs only at the start of the test, the compressor is quickly filled up with don't knows and an indeterminate signature results. A solution is to provide an operational sequence (so-called synchronizing sequence) which establishes the first test state. The GENERATOR must be initialized at the start of the test and allowed to generate a synchronizing sequence to complete the initialization of the function to be tested. The compression of responses begins only after the function has been initialized and the first test state has been established. Thus, the first

COMPRESSOR activity is delayed with respect to the first GENERATOR activity. The Mode Select & Timing Control unit must provide this delay.

4.4 Assembly of POLYDIV Modules

Implementation of a POLYDIV self-test for a single module requires two FSRs, namely, a generator and a compressor. This implementation is called Duplex POLYDIV. A system consisting of multiple modules raises possibilities of removing one of the two FSRs. The implementation which has only one FSR per module is called a Simplex POLYDIV. Figure 4.5 shows pipelined functions with Duplex and Simplex POLYDIV testing. In Simplex POLYDIV testing, the FSRs are capable of both

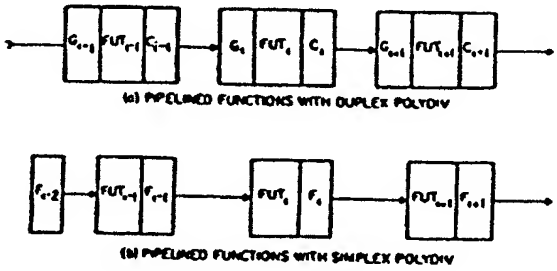

FIGURE 4.5 Two Types of POLYDIV for Pipelined Functions

generating and compressing. Their implementation is similar to the multi-mode FSR in BILBO [9]. The testing is done in two steps. First, the functions F1 and F3 are tested with FSR0 and FSR2 generating and FSR1 and FSR4 compressing. In the second step, functions F1 and F3 are tested. This time, FSR1 and FSR3 generate and FSR2 and FSR4 compress.

In the Simplex POLYDIV, each intermediate FSR tests both its upstream and downstream functions, therefore, simultaneously achieving the testing of interfaces. Duplex POLYDIV does not test the interfaces on its own. Some auxiliary means must be provided for testing the interfaces. Multi-mode FSRs in Simplex POLYDIV take more hardware than the single mode FSRs in the Duplex POLYDIV. However, since only half the number of FSRs are required, overall reduction in test hardware results. Both Simplex and Duplex POLYDIV isolate the faults to the network tested between two FSRs.

A general system may not be as simple as the pipeline just shown. It can have arbitrary interconnections between modules, with multiplexed/demultiplexed module I/Os, with I/O channel fan-ins and fan-outs and feedback around the modules (Figure 4.6(a)). Using a Simplex POLYDIV scheme (Figure 4.6(b)) in such cases can give rise to problems. First, the signature is dependent upon the test pattern generated by an FSR outside the functional module. In general applications, especially at the board level assembly, the upstream module may not have been identified at the time of POLYDIV self-test design. This requires the designer to provide for different application dependent signatures. The other alternative may be to perform signature verification away from all the functional

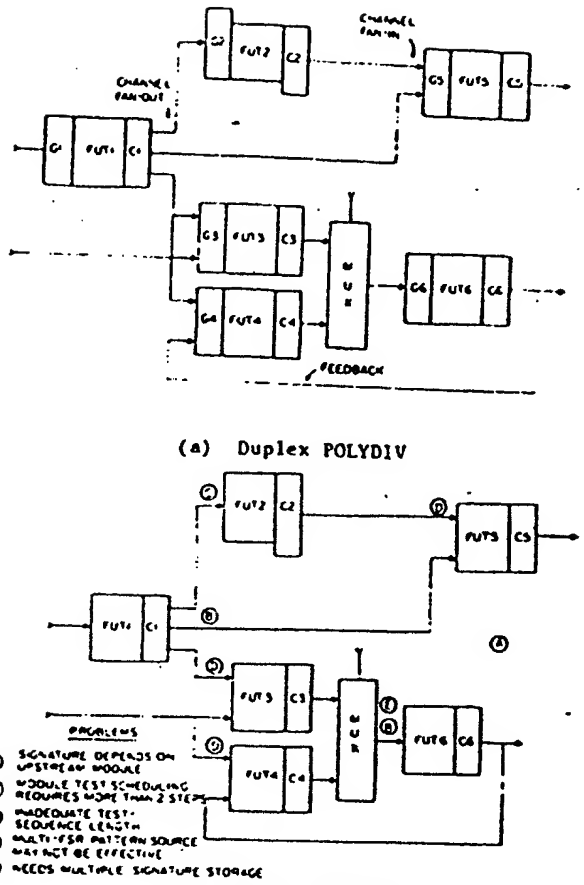

FIGURE 4.6 Multi-module Network With Arbitrary Interconnections

modules by a central signature processor. FSRs which have built-in chained serial register modes (such as in BILBO) facilitate the load/unload of signatures for this purpose.

Multiplexers in module interconnections raise a similar type of problem. Each compressor normally has a single fault-free signature attached to it (for its upstream function). When multiplexers are present, testing requires storage of multiple signatures. A central signature processor consisting of a ROM/PROM can take care of this problem.

A third source of problem arises from the channel fan-ins and fan-outs. Due to channel fan-ins, a function may receive its test vectors from more than one FSR. Even if the generating FSRs individually may have maximal length sequences, the composite test sequence may not have a maximal period. The length and the effectiveness of such a test vector set may be inadequate. Channel fan-outs create problems in test scheduling. In the pipeline, the testing is done in two steps. Each FSR generates in one of the steps and compresses in the other. A reconvergent fan-out such as the one shown in Figure 4.6 makes such two-step testing difficult (assuming that the output of FSR C2 when it is compressing cannot test effectively the FUT5 in Figure 4.6(b)). This increases the testing

time and complexity of the test scheduler.

A feedback between the modules raises similar problems. It may be advantageous to employ Duplex POLYDIV in such cases.

4.5 Function Type Dependence

PN-sequences are excellent for most types of functions, except the array type functions such as RAMs, ROMs and PLAs. The use of random patterns is not effective in detecting typical failure modes in RAMs. Any of the pattern sensitivity tests, such as March, Galpat, etc., do a better job. ROMs can be tested by POLYDIV. The address is fed by the input FSR. The ROM contents read out are compressed by the output FSR. As per Thaler's scheme [10], better fault detection is obtained by reading the ROM in different address orders. This may necessitate running the input FSR with more than one seed and may require multiple signatures.

PN-sequences generally have no difficulty in testing PLAs, except, when PLAs employ large fan-ins, the testing may require much longer sequences than usually required by a deterministic test sequence.

4.6 Assembly Level Dependence

The function under test (FUT) in all previous discussion can be a single chip or a part of a large chip. It can also be comprised of several chips and even whole boards. The assembly level of the function has some distinct implications on the POLYDIV design.

The first distinction is in the flexibility in the test design process. For chip level POLYDIV, a designer may have maximum flexibility in choosing and placing the constituents of a POLYDIV scheme, but a designer of board level POLYDIV may be constrained by the availability of particular types of chips. This also leads to the differences in the cost constraints of the different levels of assembly. For the chip level POLYDIV, the major concerns are the incremental transistor costs, increase in the chip area, and the number of pads. For the board level POLYDIV, the number of ICs, the board area used, and the pin-out are important.

Another important distinction is in the fault isolation. A POLYDIV self-testing scheme isolates faults only to the extent of network between the generator-compressor pairs used in testing. For chips with POLYDIV, this resolution is adequate since the objective usually is only to perform go/nogo testing. But, when the functions consist of several chips or boards, the resolution provided by POLYDIV may be barely adequate. Some auxiliary test means may be required to isolate to a faulty chip or a board.

There is an occasional problem in board level POLYDIV that arises when off-the-shelf ICs are used for implementation of functions. Due to unavailability of the chip's internal details, the signature obtained by software models may differ from the actual device signature. Sometimes the same IC may be received from different vendors. These ICs are usually equivalent functionally, but may not be so when tested with pseudorandom test vectors and may give rise to different signatures. The solution in both cases is to use a known good device for generating the fault-free signature. But, the fault coverage analysis may still have to rely on the modelled device.

5. FAULT COVERAGE OF POLYDIV SCHEMES

The fault coverage of POLYDIV self-testing is contributed by two factors: the basic effectiveness of the PN-sequence generated by the input FSR, and the error coverage of the output FSR. Both factors may be affected by a choice of the characteristic polynomial, the number of stages in the FSRs, and the seed. The best measure of their combined effect is the fault coverage as determined by fault simulation. The cost and accuracy of the fault coverage depends on the method of fault simulation employed.

The fault-simulation techniques known today include hardware and software fault-insertion simulators, and parallel, deductive and concurrent fault simulators. Before using any of the techniques, one must ascertain how closely the fault simulation corresponds to the method of error/fault detection by the data compressor. In actual POLYDIV testing (in field or factory), the faults remain present throughout the test run and are processed by all the test inputs, and the PASS/FAIL decision is made at the end of the test by checking the final signature. The fault-simulation technique and the model of POLYDIV employed should do exactly the same.

This condition is readily met in either of the hardware or software fault-insertion simulators. The trouble with hardware fault simulators is that they are rarely available. The software fault-insert simulators take up too much of CPU time.

A deductive fault simulator that assumes single stuck-at faults can be expected to follow the mechanism of error capture in data compressors if the model for the POLYDIV is such that it evaluates the output of the compressor only at the end of the test run. The deductive fault simulators consider all the single stuck-at faults simultaneously and, therefore, are generally faster than fault-insert simulators. However, since the compressor output is disabled until the end of the test run, it tends to cause phenomenal growth in fault lists of some of the nodes in the model. This adversely affects the execution of the fault simulation in two ways. First, if there are no internal protections in the program, the fault lists may overflow the disk storage even for a moderately sized model. Second, due to increased size of fault lists, and fault-list oscillations, processing time with each additional test vector grows exponentially. This offsets some of the possible time gain over the fault-insertion simulator.

A more economical way of testing with deductive fault simulators is to allow the model to observe all intermediate outputs of the compressor. The detection of faults is not inhibited as in previous methods, and the faults are removed from the fault lists as they get detected by the applied patterns. (Some fault-simulation systems may have a user option to control the number of times a fault must be detected before it is removed from further processing.) This limits the fault list growth and hence reduces demand on processing time and disk storage. The fault coverage predicted by this method is not accurate, because faults are removed as they get detected and are not exercised by all the test vectors. Nonetheless, if the compressor has low error escape probability, the method provides much faster results with negligible loss of accuracy in most cases. Comparative results of an experiment with the three methods just discussed are given in Table 5.1.

FAULT-SIMULATION METHOD	% FAULT-COVERAGE	TOTAL CPU TIME IN HOURS
FAULT-INSERT: (SINGLE STUCK-AT FAULT AT A TIME)	91.53	54
DEDUCTIVE: OBSERVE FINAL SIGNATURE ONLY	91.53	5.2
DEDUCTIVE: OBSERVE ALL THE SIGNATURES	91.78	0.2

Remarks: Total Number of Faults = 1229
Number of Clock Cycles = 255

TABLE 5.1 Fault Simulations of POLYDIV for 4 X 4 Bit Array Multiplier

6. SUMMARY AND CONCLUSIONS

As VLSI systems become commonplace, more and more chips will have self-test features on them. Two basic requirements of self-testing, namely, a test vector source and a response evaluator, are efficiently and economically met by FSRs in the POLYDIV schemes. The testing is done off-line at the system clock rate, and without need for generating and storing the test vectors and the test responses.

The POLYDIV technique is best applied at the chip level. Due to the regular internal structure of FSRs and flexibility in designing at the time of chip development, the technique can be implemented with low incremental transistor costs and minimal design effort. The technique can also be applied at other levels of assembly, but the cost and some of the design trade-offs are different.

POLYDIV self-testing provides an excellent go/nogo type testing, but does not permit effective fault isolation within the self-testing function. When the function is implemented with several chips or boards, some auxiliary testing means may be required to isolate faults to a faulty chip or a board.

The generality of digital networks may give rise to some design complications such as inadequate/ineffective test sequences, indeterminate signatures, multiple signatures and their storage, function type, application, and interface dependence of implementation. Each of these problems is solvable. But, due to the absence of established design rules, the solutions are not unique.

The POLYDIV self-testing is evaluated for its hardware costs and effectiveness in detecting faults. The incremental hardware cost depends on the basic function complexity and the nature of FSRs used (Table 6.1). The best available measure of effectiveness is the fault coverage obtained by single stuck-at fault simulations. The cost and accuracy of fault coverage depends on the method of fault simulation employed.

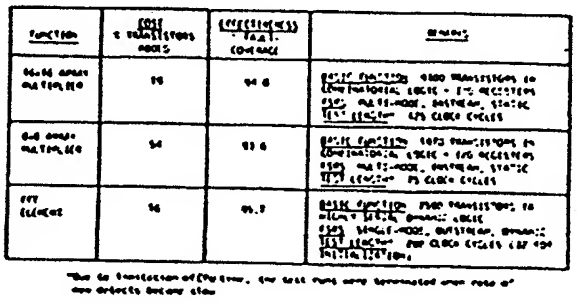

TABLE 6.1 Cost and Performance of Three Chip Level POLYDIV Schemes

Due to simplicity of design and implementation, and ability to give high performance with great convenience in testing and usage, the application of POLYDIV schemes for off-line self-testing is expected to proliferate in the VLSI era.

REFERENCES

[1] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," 14th Annual Design Automation Conference, pp 462-467, June 1977.

[2] S. Funatsu, N. Wakatsuki, T. Arima, "Test Generation System in Japan," 12th Annual Design Automation Conference, pp 114-122, June 1975.

[3] James Stewart, "Future Testing of Large LSI Circuit Cards," Digest of Papers 1977 Semiconductor Test Symposium, pp 6-18.

[4] W. W. Peterson and E. J. Weldon, Jr., "Error-Correcting Codes," The MIT Press, Cambridge, MA, 1972.

[5] S. W. Golomb, "Shift-Register Sequences," Holden-Day, Inc., San Francisco, 1967.

[6] Dennis R. Morgan, "Autocorrelation Function of Sequential M-bit Words Taken From an N-bit Shift Register (PN) Sequence," IEEE Trans. Comput., vol. C-29, no. 5, pp 408-440, May 1980.

[7] John J. Shedletsky, "Random Testing: Practicality vs Verified Effectiveness," Proc. 7th Annual Int'l Conf. on Fault-Tolerant Comp., pp 175-179, June 1977.

[8] James E. Smith, "Measures of Effectiveness of Fault Signature Analysis," IEEE Trans. Comput., vol. C-29, no. 6, pp 510-514, June 1980.

[9] B. Konemann, J. Mucha, G. Zwiehoff, "Built-in Logic Block Observation Techniques," Proc. IEEE Test Conf., pp 37-41, 1979.

[10] H. Thaler, "Pattern Verification and Address Sequence Sensitivity of ROMs by Signature Testing," Proc. IEEE Test Conf., pp 84-85, 1978.

[11] R. W. Heckelman and D. K. Bhavsar, "Miscellaneous Design Problems and Approaches With Polynomial Division (Signature Analysis)," Design for Test Workshop, Vail, CO, April 1981.

[12] R. A. Frohwerk, "Signature Analysis: A New Digital Field Service Method," Hewlett-Packard Journal, pp 2-8, May 1977.

[13] N. Benovitz, et al, "Fault Detection/Isolation Results From AAFIS Hardware Built-in Test," NAECON '76 Record, pp 215-222.

APPENDIX

Definition: If Y_i is the i^{th} response vector and P is the characteristic polynomial of the compression, then $R^t(Y_i)$ is defined as the remainder after t steps of dividing Y_i by P.

Theorem: The response compression of a sequence of L-many vectors of r-bits each by an r-bit parallel compressor (parallel linear feedback shift register) is equivalent to superposition of L-many time-staggered polynomial division processes, each initialized and triggered by a response vector. The final signature is given by

$$X = R^{L-1}(Y_1) + R^{L-2}(Y_2) + \cdots + R^1(Y_{L-1}) + R^0(Y_L)$$

Proof: The equivalent diagram of a parallel compressor in polynomial divider form is shown in Figure A1. In each clock interval, the compressor forms a new remainder by subtracting the characteristic polynomial from the old remainder and adding a new response vector. (Addition and subtraction are performed modulo-2.) This is equivalent to carrying out one step of the division process on the old remainder and adding to the remainder a new response vector. Due to the linearity of the process, the remainder in the next division step is mod-2 summation of the following:

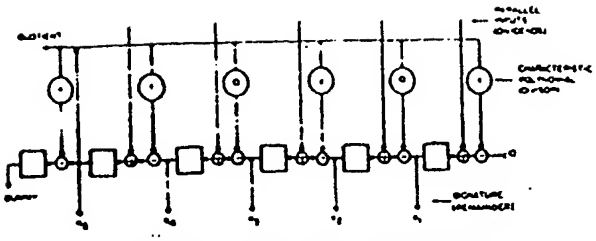

FIGURE A1 Parallel Compression

1. The remainder from the old remainder after one more step of division.

2. The remainder from the previous response vector after one step of division.

3. The new response vector.

The process is recursively carried out until all the response vectors are exhausted. The signature, therefore, is modulo-2 summation of remainders from L-many time-staggered polynomial division processes.

Theorem: Let the response sequence consist of L vectors of m-bits each. Assuming all error sequences to be equally likely, the probability of detecting errors in the response sequence by an r-bit parallel compressor is given by

$$1 - \frac{2^{mL-r} - 1}{2^{mL} - 1}$$

Proof: For L response vectors of m-bits each, there are $2^{mL} - 1$ possible error sequences. Since an r-bit feedback shift register can have only 2^r distinct states, $\frac{2^{mL}}{2^r} - 1$ error sequences in the response result in the same signature as the error-free signature, and hence remain undetected†.

† It can be shown that an r-stage linear feedback shift register used as a parallel compressor maps 2^{mL} possible sequences into 2^r signatures uniformly.